Reddit Popularity Predictor (Multi-model)

A Streamlit web application to predict the popularity of Reddit posts using multiple regression models. Users can upload their own dataset. Supports Linear Regression, Random Forest, and Gradient Boosting. Includes feature engineering, visualization, and manual prediction options. This will also predict the score of the popularity if given the values of the

Team and Collaborators

- Team ID: 12
- Team Members: Deesha C, Chitra Madarkhandi

Prerequisites:

1.Python: 3.8 or higher
2. Install required libraries:
pip install streamlit pandas numpy scikit-learn matplotlib joblib wordcloud
3.CSV Inputs: train.csv, val.csv, test.csv with columns like ups, downs, body, created_utc, score.If not provided, the app auto-generates sample data.
4. Run the app:
streamlit run ui.py

Installation:

1.Clone the Repository
Start by cloning the project repository to your local machine using git.
git clone: 12_ML_Reddit_Post_Popularity/ at main · PES2UG23CS165EC/12_ML_Reddit_Post_Popularity
cd your-repository-name

Python 3.8+
https://www.python.org/downloads/
Install dependencies:
pip install streamlit pandas numpy scikit-learn matplotlib joblib wordcloud
Libraries:
streamlit
pandas
numpy
scikit-learn
matplotlib
joblib
wordcloud

Usage:

1. Data Preparation
Upload your own train.csv, val.csv, and test.csv files containing columns like ups, downs, body, created_utc, and score.
If no files are uploaded, the app auto-generates a synthetic dataset.

2. Run the Streamlit App
streamlit run app.py

3. In the GUI

- Upload Data**:** Add your CSVs (optional).
- Select Models: Choose Linear Regression, Random Forest, or Gradient Boosting.
- Train & Evaluate: Click “Train & Evaluate” to see MSE/MAE scores, plots, and feature importance.
- Predictor: Enter custom values to predict popularity scores using one or all models.
- Reset: Use “Finish / Reset app” to clear all models and start fresh.

Methodology

Data Preprocessing

- Clean and prepare the dataset using preprocess_basic() to compute key features like body_length and post_hour.
- Handle missing values and ensure numerical consistency.
- Define the target variable score (popularity).

Feature Engineering

- Use add_engineered_features() to create derived metrics such as ups_downs_product, ups_squared, ups_to_downs, and body_hour_interaction.
- Model Training & Validation
- Train multiple regression models — **Linear Regression**, **Random Forest**, and **Gradient Boosting** — on the training set.
- Evaluate models on the validation set using **Mean Squared Error (MSE)** and **Mean Absolute Error (MAE)**.

Testing & Visualization

- Assess each trained model on the test set and plot **Actual vs Predicted** results.
- Display combined **feature importance** (top 10) for tree-based models.
- Generate a **word cloud** of high-scoring posts to visualize common terms.

Manual Prediction

- Allow users to input post features manually and predict popularity using one or all trained models interactively within the Streamlit app.

Model Persistence & Reset

- Save the best-performing model (.joblib file) for reuse.
- “Finish / Reset app” clears session state and restarts the interface for a new run.

Dataset Requirements

Expected columns:
- ups
- downs
- body
- created_utc
- score

Modeling Details

- Base features: ups, downs, body_length, post_hour
- Engineered features: ups_downs_product, ups_squared, body_hour_interaction, ups_log,downs_log, ups_to_downs

Visualizations

- Actual vs Predicted scatter plots (test set)
- Feature importance table (top 10) from tree-based models
- Word Cloud of high-scoring posts (requires wordcloud)

Manual Prediction

- Input ups, downs, body_length, post_hour
- Select a trained model or predict with all
- Predictions shown in table or individually

Reset / Finish

- Clears session state, deletes trained models, scalers, predictions, and visualizations
- Returns app to initial state